REPORT DOCUMENTATION PAGE

FORM APPROVED
OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
19-11-2002	Final report	6/1/97 – 7/14/00

4. TITLE AND SUBTITLE

Natural Interaction with Pedagogical Agents in Virtual Interactions

5a. CONTRACT NUMBERS

N00014-97-1-0598

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

W. Lewis Johnson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

USC INFORMATION SCIENCES INSTITUTE
4676 ADMIRALTY WAY
MARINA DEL REY, CA 90292-6695

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 N. Quincy St.
Arlington, VA 22217-5660

10. SPONSORING/MONITOR'S ACRONYM(S)

ONR

11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT

UNCLASSIFIED/UNLIMITED

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The USC / Information Sciences Institute (ISI), in collaboration with Lockheed Martin and USC Behavior Technology Laboratory, conducted a research project named Virtual Environments for Training (VET) between October 1995 and February 1999. VET developed an integrated prototype of an intelligent virtual reality-based training system. ISI's main contribution to this project was the development of the Steve pedagogical agent, an embodied agent that can participate in training simulations, as a virtual coach or a virtual team member.

In support of this grant ISI received a supplemental AASERT award, entitled "Natural Interaction with Pedagogical Agents in Virtual Environments." Funding for this work began in June of 1997 and continued through August of 2000. The overall purpose of this project was to improve the naturalness and effective of interaction between humans and the Steve pedagogical agent, both during lesson authoring and lesson execution. Several students were supported by this award. This report summarizes the work of each student who was supported by the grant.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
			W. Lewis Johnson

106 075

a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (Include area code)
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			310-448-8210

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

Natural Interaction with Pedagogical Agents in Virtual Environments

W. Lewis Johnson, Principal Investigator
ONR Grant No. N00014-97-1-0598
ISI Proposal No. 96-ISI-104

The USC / Information Sciences Institute (ISI), in collaboration with Lockheed Martin and USC Behavior Technology Laboratory, conducted a research project named Virtual Environments for Training (VET) between October 1995 and February 1999. VET developed an integrated prototype of an intelligent virtual reality-based training system. ISI's main contribution to this project was the development of the Steve pedagogical agent, an embodied agent that can participate in training simulations, as a virtual coach or a virtual team member.

In support of this grant ISI received a supplemental AASERT award, entitled "Natural Interaction with Pedagogical Agents in Virtual Environments." Funding for this work began in June of 1997 and continued through August of 2000. The overall purpose of this project was to improve the naturalness and effective of interaction between humans and the Steve pedagogical agent, both during lesson authoring and lesson execution. Several students were supported by this award. This report summarizes the work of each student who was supported by the grant.

Richard Angros's Diligent

Richard Angros conducted his dissertation research on the topic of authoring knowledge for use by the Steve agent in training. The main results of his work were published recently at the First International Joint Conference on Autonomous Agents and Multi-Agent Systems,[i] and summarized below.

Angros's system, called Diligent, acquires subject matter knowledge from instructors using a method inspired by human tutorial dialog, combining direct specification, demonstration, and experimentation. The human instructor demonstrates the skill being taught, while the agent observes the effects of the procedure on the simulated world. The agent then autonomously experiments with the procedure, making modifications to it, in order to understand the role of each step in the procedure. At various points the instructor can provide clarifications, and modify the developing procedural description as needed.

Types of acquired knowledge

Diligent acquires three types of domain knowledge: (1) task models, which describe domain procedures, (2) operators, which model the preconditions and effects of domain actions, and (3) linguistic knowledge, which allows a tutor to talk about the domain.

The representation for task models must satisfy two requirements. First, it must allow the tutor to determine the next appropriate action while demonstrating a task or watching a

student, even if this requires adapting standard procedures to unexpected student actions. Second, it must allow the tutor to explain the role of any suggested action in completing the task.

To meet these requirements, Diligent uses a relatively standard hierarchical plan representation to formalize tasks. A task model includes a set of steps, a set of end goals, a set of ordering constraints, and a set of causal links. Each step may either be a primitive action (e.g., press a button) or a composite action (i.e., itself a task). The end goals of the task describe a state the environment must be in for the task to be considered complete (e.g., "valve1 open" or "engine on"). Ordering constraints impose binary ordering relations between steps in the procedure. Finally, causal links represent the role of steps in a task; each causal link specifies that one step in the task achieves a particular precondition for another step in the task (or for termination of the task). For example, pulling out a dipstick achieves the goal of exposing the level indicator, which is a precondition for checking the oil level.

This representation was chosen for its compatibility with relevant prior work. For simulation-based tutoring, Rickel and Johnson have shown how partial-order planning techniques can use this task representation to construct and revise plans for completing a task in the face of unexpected student actions, as well as dynamically generate explanations for the role of each step in the plan in completing the task. The causal links in the representation are especially important. They support plan revision by allowing the tutor to determine which task steps are still relevant to achieving the task's end goals. They also support explanation generation; the tutor can combine its knowledge of the causal links in the task model with its knowledge of which parts of the task are still relevant in order to explain the role of an action in completing the task.

To complement the task models it acquires, Diligent also acquires a set of operators that model the preconditions and effects of the domain actions in the task models. Task models only specify those preconditions and effects of task steps that are relevant to completing the task. In contrast, operators specify all the possible effects of a domain action, and the required preconditions for each effect. Like most work on planning, our work focuses on actions with discrete (rather than continuous) effects.

Finally, to allow the tutor to talk to students, Diligent acquires linguistic knowledge. Specifically, Diligent currently learns text fragments for the various elements of its ontology; these can be used with domain-independent text templates to support natural language generation during tutoring. In the future, we plan to support the acquisition of more structured linguistic knowledge to support more sophisticated natural language generation techniques.

Sources of knowledge

In simulation-based training, students learn procedural tasks by practicing them in a simulation of their work environment. The simulator has a user interface that allows students to perform domain actions and see their effects. Diligent exploits the same simulator to acquire its domain knowledge.

Diligent makes few assumptions about the simulator. First, it assumes that the human instructor can perform domain actions, so that the instructor can demonstrate tasks for Diligent. Second, it assumes that Diligent can perform the same actions itself by sending commands to the simulator; this enables it to experiment with domain tasks. Third, it assumes that the simulator will send Diligent an *action observation* whenever Diligent or the human instructor performs an action. The action observation should include the simulation state in which the action was performed, the action that was taken, and its effects. For compatibility with a wide range of simulators, Diligent assumes that the simulation state is represented as a set of state variables and their values; an effect is a new value for a state variable. Finally, Diligent requires the ability to save a state of the simulator and restore it later, so that it can repeat experiments from a known state.

Unlike some other systems, we do not assume that Diligent can make arbitrary changes to the simulation state. While the ability to make arbitrary changes greatly simplifies learning, allowing the agent to see the effect of each change on an action's effects, this ability is impractical for most simulators. Many simulators enforce a wide variety of constraints among state variables. While they are able to maintain these constraints for legal domain actions, they often cannot maintain them in the face of arbitrary changes to individual state variables. Even if they could, propagation of these constraints would violate the agent's desire to make individual changes.

It may seem as if a computer tutor could simply use the domain logic in the simulator, rather than acquiring it indirectly by learning from the simulator. However, this is also impractical. A wide variety of simulators exist, each with its own (often custom) representation. Rather than build tutoring methods that can exploit these different representations, or build methods that can automatically convert from these representations into our target representation, our goal is to build general learning methods that can acquire our target domain knowledge from any simulator that supports the API described above.

In addition to the simulator, Diligent has a second source of knowledge: the human instructor. Our research focuses on human instructors that have domain expertise but not necessarily any abilities to program or formalize knowledge. Thus, teaching Diligent should be, as much as possible, like teaching another person. The instructor teaches Diligent by demonstrating tasks in the simulator and by using a GUI to issue commands, directly provide elements of domain knowledge, answer Diligent's questions, and review Diligent's knowledge.

Integrating demonstration, experimentation, and direct specification

Diligent places relatively few burdens on instructors, compared to previous systems. Some earlier systems ask a large number of questions, some require a large number of demonstrations, and some require an initial domain theory. Diligent minimizes the need for demonstrations, questions, and an initial domain theory through a novel combination of programming by demonstration, autonomous experimentation, and direct specification.

Demonstrations

Diligent begins learning about a procedure through a process of programming by demonstration. The human instructor issues a command for Diligent to observe his actions, and then performs the procedure by taking actions in the simulator. During this demonstration, Diligent receives action observation messages from the simulator, processing them to create operators, and stores the sequence of actions as a demonstration.

Diligent records actions as a human instructor executes each step in the task being learned, noting the state of the simulated environment before and after each action. For each action observed, as described in the next section, an operator is created to model the effects of the action; each step in the demonstration is recorded as an instance of an operator with a set of particular effects.

By the time the demonstration is complete, Diligent has learned a sequence of steps that can be used to perform the task. This provides Diligent with the set of steps required for its target task model, as well as one possible ordering of those steps; at this point, as far as Diligent knows, other orderings may also work. To establish the end goals of the task, the second part of our task representation, Diligent hypothesizes that likely goals are the final values of state variables that changed value during the demonstration. The instructor is then allowed to review this list and remove goals that are merely side effects of the task.

At this point, Diligent could derive the ordering constraints and causal links from its operator models, but there would most likely be errors. During a demonstration, Diligent typically only sees an action taken under one set of circumstances. This is insufficient to identify the precise preconditions of operators, which are needed to identify the ordering constraints and causal links among steps. To refine how operators model the effects of actions, the system needs to see the actions performed under different circumstances. To produce action observations with different preconditions than seen during the instructor's demonstration, Diligent experiments with the task in the simulated world.

Experimentation

Diligent uses the initial demonstration to generate experiments. The experiments are conducted by repeating the task once for each step in the original demonstration. Each time through, Diligent omits a different step to see how the absence of that step affects subsequent steps. These experiments are a focused way for Diligent to learn on its own with a minimal amount of background knowledge of the task and environment.

Learning is performed by refining the preconditions of operators associated with each action using a version space algorithm. A version space maintains bounds representing the most specific (s-set) and most general (g-set) combinations of preconditions possible for each effect of the operator. The state of the world before each action, and the changes that occur afterwards, are used to create new operators and update old ones. To reduce version space memory requirements, Diligent use the INBF algorithm in which the s-set and g-set are each represented by a single conjunctive condition. Initially, the s-set is the

state before the action and only matches that state, while the g-set is empty and can match any state. Successful applications of old operators under new conditions can be used to broaden the s-set by removing preconditions. Conversely, actions that fail to replicate the effects of an old operator may be useful in narrowing the g-set by adding additional preconditions. The combination of the instructor's original demonstration and Diligent's own experiments provide a range of examples of each operator, and the version space representation represents the conclusions Diligent has drawn about these operators and any remaining uncertainty due to insufficient examples.

Direct Specification

After the system has finished experimenting, it can use the original demonstration, the end goals of the task, and its refined operators to generate a representation of the learned task. Recall that the record of the demonstration contains the state before and after each step. Since Diligent associates an operator with each step in the demonstration, it can use its model of that operator to identify the preconditions that were required in the state before the step in order to produce the effects that were observed after it was executed. This allows Diligent to augment the demonstration to include the preconditions of each step as well as its effects. Given the initial state, Diligent can then analytically derive how the initial state and earlier steps establish both end goal conditions and preconditions for later steps. This allows Diligent to identify the causal links and ordering constraints of the task.

The instructor can review the knowledge Diligent has learned by either examining a graph of the task model or by allowing STEVE to demonstrate its understanding of the task by trying to teach it back to the instructor. During this review, the instructor can use GUIs to refine Diligent's task model by adding or removing steps, ordering constraints, and causal links. Such modifications may be necessary if Diligent did not execute an action in a wide enough variety of circumstances to fully learn the preconditions of its effects.

Heuristics for Refining Operators

The g-set and s-set in Diligent's operator models represent conservative, provably correct bounds on the true preconditions (given the assumption of conjunctive preconditions). When they converge, Diligent knows the true preconditions. Before they converge, Diligent has a representation of its uncertainty.

In order to derive a task model and display it to the instructor before the operator models have converged, Diligent must make a reasonable guess at the true preconditions given the current bounds. The system addresses this problem by using heuristics to create and maintain a working set of likely preconditions (h-set). The heuristics are designed to add a small number of preconditions to the g-set that hopefully cover all or most of the actual preconditions.

The likely preconditions (h-set) are currently identified using two heuristics:

- **Earlier steps establish preconditions for later steps.** The steps in a demonstration are related, and the instructor probably has reasons for

demonstrating steps in a given order. A likely reason is that the state changes of earlier steps establish the preconditions of later steps. In this case, Diligent uses as preconditions the values of state variables that were changed by earlier steps. The system ignores possible preconditions that were true before the procedure started because only state variables that change value can differentiate between orders of steps that achieve the goal state and those that do not.

- **Focus on state variables that change value.** If a step changes a particular state variable as one of its effects, the previous value was probably a precondition.

The heuristics are only used to initialize the h-set of each operator. As Diligent sees additional examples of an operator, the h-set is refined with the inductive version space techniques described above, so the h-set is guaranteed to lie between the g-set and s-set bounds.

Example

To illustrate these concepts, we will discuss authoring a very simple task. Assume that Diligent has no prior knowledge of how the demonstration's steps will affect the simulation.

1. The human instructor tells Diligent that he wants to author a new procedure, and he provides a name ("shut-valves") to identify the procedure.
2. The instructor demonstrates the procedure by using the mouse to manipulate objects in the simulation's graphical window. This particular demonstration consists of selecting (i.e., clicking on) three valves sequentially: `valve1`, `valve2`, and `valve3`. When the instructor selects a valve, the valve closes.
3. After the demonstration, the procedure's goals must be specified. Diligent hypothesizes that likely goals are the final values of state variables that changed value during the demonstration: `(valve1 shut) (valve2 shut) (valve3 shut)`. In this case, the instructor verifies that these are the correct goals.
4. Now the instructor tells Diligent to experiment with the procedure. While Diligent could generate a task model without experimenting, the task model would contain errors because Diligent does not yet understand the preconditions of each step (e.g., preconditions for shutting `valve1`). The experiments attempt to understand if and how each step depends on earlier steps. In this example, Diligent will repeat the procedure twice; the first time it skips step one (closing `valve1`) to see the effect on steps two and three, and the second time it skips step two (closing `valve2`) to see the effect on step three.

To better understand why these experiments are useful, consider the operator that represents shutting `valve3`. When creating an operator during the demonstration, Diligent hypothesizes that state changes from earlier steps are likely preconditions (h-set). The version space g-set is empty because no preconditions have been shown to be necessary, while the version space s-set contains many irrelevant conditions. In contrast, the h-set is more focused on how the demonstration manipulates the simulation's state.

The demonstration's three steps are independent and could correctly be performed in any order. However, because Diligent uses likely preconditions (h-set) to generate the task model, Diligent initially believes that `valve1` needs to be shut before `valve2`, and `valve2` needs to be shut before `valve3`. Diligent's experiments correct this problem, and, after the experiments finish, Diligent can build a correct task model.

Extensions

To allow Diligent to model more complex procedures and to support better nteraction with the instructor, several extensions to the above mechanisms have been implemented. One extension is support for hierarchical task representations, where a subtask is treated as a single step in a larger task. This allows the reuse of existing tasks and improves scalability because large, complicated tasks can be broken into simpler subtasks. An instructor can specify a subtask during a demonstration either by defining a new task as a step in the current procedure or by indicating that an existing task should be performed at that point.

To treat a subtask as a single step, the internal details of the subtask are ignored and only its preconditions and effects are considered by the parent task. To achieve a subtask's desired effects, Diligent uses a version of STEVE's partial-order planning algorithm for deciding how to perform a subtask. Diligent uses the likely preconditions (h-set) of the subtask's steps to determine ordering constraints, causal links and which steps to perform. However, Diligent will execute all subtask steps that achieve necessary effects, even if preconditions do not appear to be met, because Diligent's knowledge of the subtask preconditions might be incomplete or incorrect.

Besides hierarchical tasks, Diligent also supports information gathering or sensing actions. A *sensing action* gathers information about the environment's state without changing it (e.g., looking at a light). Because sensing actions do not change the state and might be performed at any time, there needs to be a mechanism to insure that they are performed in the proper context (e.g., look at the light when the motor is off). For sensing action preconditions, Diligent uses the values of the state variables that have already changed value in the current task. To record that a sensing action has been performed in the proper context, Diligent creates internal "mental" state variables, which are not present in the environment's state. To require that sensing actions are performed, the mental effects are added to the task's goals.

The above discussion only mentioned a single demonstration of each task. However, Diligent allows instructors to provide multiple demonstrations of tasks. This allows instructors to iteratively refine task models, correct mistakes, or elaborate on subtasks. However, multiple demonstrations raise issues of which initial state to use for each step when deriving causal links and ordering constraints. Diligent's algorithms for processing multiple demonstrations provide a foundation and are sufficient for the relatively short procedures that we have looked at. Incorporating algorithms for more complicated domains is an area for future work.

Evaluation

We evaluated Diligent to determine whether it simplifies the job of authoring procedural knowledge. Since Diligent employs two techniques to assist instructors, namely demonstrations and experiments, we evaluated their separate contributions in facilitating authoring. Our hypothesis was that both demonstrations and experiments would reduce the effort required by the instructor and result in more accurate task models. To test the hypothesis, fifteen computer science graduate students were divided into three groups that authored procedural knowledge differently: group G1 (four subjects) used demonstrations, experiments, and direct specification; group G2 (six subjects) used only demonstrations and direct specification; and group G3 (five subjects) used only direct specification.

Two dependent measures were used to assess ease of authoring and quality of the resulting knowledge base. Edits is the number of deliberate changes made to the knowledge base (e.g., demonstrating a step or changing a precondition). Errors is the number of mistakes in the task model, either missing or spurious ordering constraints and causal links. These errors are caused by missing, unnecessary and incorrect preconditions and goal conditions. The sum of the edits and the errors metrics, called total required effort, estimates the total work required to create a correct task model.

For each subject, the evaluation covered two consecutive days. The subjects were trained for approximately 2 hours the first day and had a 30 minute review on the second day. After training, the subjects authored two machine maintenance procedures (proc1 and proc2) of differing complexity, proc1 the more complex of the two. Both procedures were authored with an initially empty knowledge base. While a subject was authoring, the system collected data about his activities. Afterwards, the resulting task models were manually compared against the desired task models. When authoring each procedure, subjects were given a time limit, which many subjects reached. Reaching the time limit was likely to reduce the edits and increase the number of errors.

The authoring was an iterative, multistage process. Subjects were given functional descriptions of the procedures that gave them enough information to reconstruct the formal details (e.g., steps, ordering constraints, and causal links) without specifying them directly. After studying the functional descriptions to understand the procedures, they used Diligent to specify the steps through either demonstrations (G1 and G2) or direct specification (G3). After demonstrating, G1 subjects could refine the initial task model by asking Diligent to experiment with the demonstration. All subjects could then edit the task models and test them with the STEVE tutor. The process was iterative because subjects could add more steps, experiment again, perform additional editing, and retest the task models.

The following are some of the main trends that emerged in the evaluation. For proc2, the groups were significantly different (ANOVA) and group G2 (only demonstrations) is significantly better than G3 (only direct specification). Next, consider errors. For proc1, the groups have a significant difference (ANOVA) for both errors of omission (.001 probability) and total errors. For errors of omission on proc1, group G2 is significantly

better than G3. Finally, consider total effort. Both procedures have significant ANOVA for total effort, and for proc1, group G1 (experiments) is significantly better than G2 (only demonstrations). Overall, though, the effects with the complex procedure, proc1, are much greater than they were for proc2, suggesting that Diligent's assistance is most beneficial with complex procedures that have significant opportunities for error.

The results suggest that demonstration and experimentation both played a role in improving the quality of the task models and reducing the work required to create them. Task models acquired through demonstrations typically contained spurious elements that the instructors then needed to delete; however it was easier for them to recognize spurious elements than to notice when elements were missing, as group G3 was required to do. Experimentation eliminated many of the spurious elements, further reducing the effort involved. These effects were most prominent in the more complex procedure containing many elements.

Summary

The Diligent method exploits the structure and behavior of the simulation environment, by observing the effects of procedures on the state of the simulation. It thus takes advantage of infrastructure that must be created in any case, since agent-enhanced simulation environments require interactive simulations that the agents are able to observe and interact with. Demonstration and experimentation provide opportunities to employ machine learning techniques to reduce the effort required to author procedural knowledge. The agent is able to acquire a significant amount of knowledge from a small number of demonstrations, and the human instructor can further refine the agent's knowledge through interactive testing and direct specification.

This method is best suited for authoring procedural skills where the primary focus of the procedure is to achieve some desired effect in the virtual world, and where the consequences of actions are readily observed. Not all procedural skills fit this description. For example, we have chosen not to apply the Diligent method to the acquisition of clinical procedures for the Adele pedagogical agent. Although diagnostic procedures in medicine have sequences of steps, the rationales for the steps are often the conclusions that they permit about the patient's condition rather than the effects they have on the patient. Furthermore, any effects on the patient are likely to be indirect and not immediately discernible. Diligent's experimental method is of more limited value in domains such as this, although demonstrations might still perform a useful role in describing procedures.

Arthur Kroetz

For his Ph.D. research, Arthur Kroetz, a student in the education program, evaluated the Steve virtual tutor, using a single case study method with qualitative measures. Relevant publications, personal interviews, system documentation, the program environment, and the simulation procedure itself were examined. The simulation procedure studied 10 non-randomly chosen subjects interacting with Steve. Subjects observed the Steve tutor demonstrate the readiness procedure and then had to replicate the procedure themselves. Three constructs were chosen to elucidate the factors of animated character enhancement:

tutor believability, active participation by the subject (constructivism), and what kind of mental models of the simulation were produced. Results of the experiment showed that even though Steve lacked emotions, subjects deemed him believable to the level of being a sentient humanoid. Subjects also expressed freedom to complete the procedure in a natural, engaging manner but felt constrained to touch other controls not shown in the demonstration. The mental models formed by the subjects showed lack of ability to remember details of the simulation and what was the overall function of the simulation. The presence of Steve within the simulation elicits human-lie responses from the subjects, allows for efficient learning of a complex procedure, and provides for situated, active participation in a procedural learning environment.

Andrew Scholer

Steve makes use of many of the opportunities for learning that a dialog between human and an embodied agent in a virtual world affords. Significant improvements however are needed to allow Steve to forge demonstration, experimentation, and instruction into a cohesive dialog that uses verbal and non-verbal communication. Andrew Scholer, a Ph.D. student in computer science, developed a model for human-agent tutorial dialog, and commenced implementation of this model.

The model builds upon previous work on computational models of human collaborative dialog, such as those of Rich, Sidner, Traum, and Lochbaum, extending them to account for the nonverbal interactions between collaborators in a shared environment. For this to happen, the agent's application of communication modalities such as verbal instruction, physical examples and gestures all had to be made more flexible.

The Diligent model of knowledge acquisition prevented the smooth interleaving of different modes of communication. A human instructor was able to directly provide only some of the types of information that Steve needs. Even for many of these types of knowledge, direct instruction was possibly only when Steve asked a question about a subject. For example, there was no means for in instructor to provide the name for an object other than to manipulate it and wait for Steve to ask for the object's name. To foster a dialog, the agent must be able to accept instruction about any type of knowledge it possesses and be a more active learner by demanding such instruction when it can identify gaps in its own knowledge. Furthermore, such interaction should be able to happen at any time. Although it is reasonable to make expectations based on context, and require the instructor to adequately establish context for switches in focus of dialog, context should not be enforced by limiting what a person can discuss with the agent at any given time.

Once the agent can use physical actions, instruction and gesture in a flexible manner, it will be possible to build a more natural and powerful dialog for instruction. Even abilities such as watching demonstrations and performing experiments, that are already well developed in Steve, will profit from the ability of an instructor to mix instruction modalities at will.

As an example, consider demonstrations performed by Steve. The agent watches the instructor's actions to learn about both physically performing steps and the rules governing the operators that those steps apply. Currently, there is not way for the instructor to focus Steve's attention on any particular feature of the world—a very natural process in instructing humans. If the instructor and agent were able to mix verbal and non-verbal instruction into the demonstration, the instructor would be able to point out through words and gestures the features of the world worth particular attention. By telling Steve "Watch that light" and gazing towards a particular one, the instructor should be able to specify that Steve evaluate an action solely based upon the effect that it has on that particular light.

Not only would the instructor be able to use an improved dialog with Steve to focus what the agent reasons about, but also to suggest what it should be doing. While the agent experiments with a task, the instructor should be able to focus the agent's experimentations by suggesting particular steps or relations to consider. If explicitly told that a set of steps could be performed in any order, Steve could forego the need to perform experiments and test the ordering constraints between those steps, using the instructor's advice to update its knowledge of the relationships between those steps. Such tailored instruction would be useful in Steve both in its own learning and in guiding what it should emphasize when later on it instructors human students.

Ben Moore

Speech and auditory communication play important roles in communication with Steve in the virtual environment. An undergraduate at USC, Ben Moore, implemented key elements of the VET architecture dealing with sound and speech communication. One of these components, RecAppl, used Entropic Research's speech recognition API to recognize student requests and speech acts for team members. Another component, called SoundServer, provides audio effects for objects. It acts as a service, responding to requests for audio effects. SoundServer was prototyped as a Java application.

---

[i] Angros, R. Jr., Johnson, W.L., Rickel, J., & Scholer, A. (2002). Learning domain knowledge for teaching procedural skills. Proc. of the First. Intl. Joint Conference on Autonomous Agents and Mult-Agent Systems, 1372-1378. New York: ACM Press.

UNIVERSITY OF SOUTHERN CALIFORNIA  INFORMATION SCIENCES INSTITUTE
4676 Admiralty Way Marina del Rey, CA 90292

Scalable Coordination Architectures For Deeply Distributed Systems (SCADDS)
Quarterly R & D Report No. 11
Period: 1 July 2002 – 30 September 2002

Defense Advanced Research Projects Agency (DoD)
Information Technology Office (ITO)
Under Grant # DABT63-99-1-0011
Issued by Directorate of Contracting
Fort Huachuca, AZ

Scalable Coordination Architectures for Deeply Distributed Systems Quarterly Report 1 July 2002 - 30 September 2002

1. Overview

The SCADDS project is exploring and demonstrating scalable coordination mechanisms for deeply distributed and dynamic systems. Nodes in these systems will be heterogeneous, having a range of sensing, actuation and communication capabilities. These systems raise many challenges for distributed system and network design. The first is a shift from node-centric, to data-centric network architecture. Both scalability and long lifetime call for extensive processing of data within and among the nodes of the sensor network. Rather than streaming all sensor readings back to a central site for processing, nodes autonomously exchange data, filter out uninteresting events, and identify patterns of interest. The second challenge is to build systems that are truly self-configuring; able to adapt efficiently to ad hoc deployment and both environmental and network dynamics. This paradigm shift requires a new network architecture. We are investigating an approach we call directed diffusion. We divide our report into the following subject areas:

Section 2. Diffusion Architecture
Section 3. A New Radio Communication Stack on Mica Motes
Section 4. Testbed Development
Section 5. Sensor IT Collaboration/Integration
Section 6. Administrative (Staffing, Travel, Publications)

2. Directed Diffusion Architecture

2.1 Geographical and Energy Aware Routing and Quality of Task

Previously, we tested Geographical and Energy Aware Routing (GEAR) in a high level simulator, which ignored many low level details. In order to test our protocol with more realistic lower layer details, we implemented and evaluated our protocol in ns-2. In the ns simulation, we validated our two major design choices of GEAR, i.e., load balancing and recursive forwarding inside the target region.

To evaluate load balancing in GEAR, we compare GEAR with GPSR, where next hop selection is completely based on geographical distance, therefore no load balancing is involved. In the traffic scenario we tested (1 traffic pair with source and target randomly selected, which stands for non-uniform traffic distribution), GEAR can deliver 20% to 60% more packets than GPSR before the network is partitioned. Moreover, compared to GPSR, GEAR extends network lifetime in up to 150% for small packets (180B).

Load balancing in GEAR maximizes its gain with small packets as explained. The threshold triggered beacon update will generate more overhead than GPSR. On the other hand, in order to do perimeter forwarding, GPSR carries extra fields in its header. For small packets, GEAR's extra

beacon overhead and GPSR's packet overhead will cancel each other out, so that load balancing in GEAR will show its gain. However, for larger packets, the extra header fields in GPSR can be ignored, thus the beacon overhead will offset the benefits of load balancing in GEAR. However, we expect sensor network traffic to consist of mostly packets of up to a few hundred bytes due to its energy constraints. Furthermore, the beacons in the routing layer can be reduced by piggybacking them in data packets or in MAC layer control packets.

We also use ns simulations to study if recursive forwarding provides any gain over controlled flooding in the target region and how this gain changes with packet length. Unicast packet has RTS/CTS/ACK broadcast packets overhead. However, if packets are large enough, so that control overhead can be ignored, unicast recursive forwarding is still more efficient than flooding in the target region. Simulation results show that recursive forwarding delivers from 125% to 150% more packets than controlled flooding inside the target region. When MAC control overhead is ignored (large data packets), recursive forwarding exhibits major gains over controlled flooding.

2.2 Geographical and Energy Aware Routing with PUSH

During this quarter, Fabio Silva worked on integrating our GEAR algorithm (Geographical and Energy Aware Routing, developed by Yan Yu) with PUSH. We believe the addition of GEAR support for PUSHed messages will greatly improve network lifetime when using PUSH. Exploratory Data messages will benefit from informed neighbor selection, being routed toward a target region (as opposed to being flooded in the network). This mechanism is similar to Interest messages being routed by GEAR toward the target region in Diffusion. The latest release of Diffusion, containing PUSH and GEAR, along with several other features, is currently available at both the BBN website (http://dstl.bbn.com) and at the SCADDS website (http://www.isi.edu/scadds).

During the next quarter, plans include (subject to availability of funds):

a) Continue the development and evaluation of different diffusion variants. We believe that the addition of several variants to the diffusion protocol family will allow us to better match specific application needs with more efficient communication protocols.

b) Continue to expand our experience with nested-query style applications for Diffusion. During the next quarter, we expect to develop and deploy a 24/7 application, which will track people movement on ISI's 11th floor's hallways. In addition to gaining experience in distributed application design using in-network processing, we expect to increase our understanding of Directed Diffusion protocols' behavior in a real testbed.

3. A New Radio Communication Stack on Mica Motes

In this quarter, Wei Ye worked with Deborah Estrin and John Heidemann to port S-MAC to the Mica motes. As a result, we have built a new radio communication stack that includes much more work than the MAC layer itself. We have investigated the issues of designing a layered architecture on the tiny resource-constrained sensor nodes.

The layers are intended to provide standard interfaces and services, so that various protocols can be developed in parallel. The goal of the stack design and implementation is to provide a flexible architecture that allows protocols at different levels to be easily built and compared with other similar protocols. We have implemented the physical layer and the MAC layer in our stack. Compared with the communication stack that comes with the pre-NesC release of TinyOS, our stack has some new features listed as follows. They are desirable for building different protocols and applications.

a) Different layers/components are free to define their own packet formats. They can freely add their own headers to packets from upper layers without causing interference.

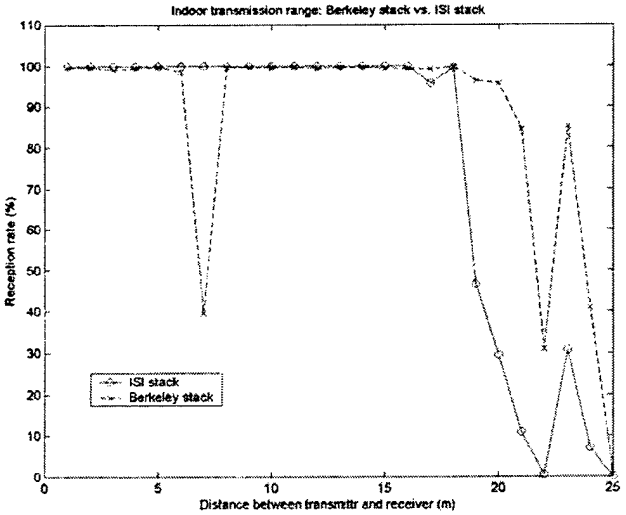
b) Packets with dramatically different lengths in fast consecutive transmissions can be reliably received. The current supported maximum packet length is 250 bytes.

c) A full-featured MAC protocol for sensor networks, S-MAC, is implemented on the stack. It provides advance features such as effective collision and overhearing avoidance, reliable and efficient transmission of long messages (can be much longer than 250 bytes), and low duty-cycle operations on radio.

For detailed design and implementation, please refer to the documentation of the communication stack at http://www.isi.edu/scadds/papers/commstack.pdf.

We have collected some performance data using the Mica motes with a 433MHz radio and matching whip antenna within the building of ISI. Results show that our stack achieves similar reliability as Berkeley's stack. Figure 1 is the reception rate of receivers at different distances to a common sender. Our stack obtains almost all 100% reception rates on the receivers whose distance are between 1m to 16m.

Jerry Zhao implemented several components in TinyOS to support SMAC in legacy applications. A SMAC generic communication component is developed with the same interfaces to TinyOS applications as for original MAC. A generic base transceiver component interfaces SMAC with PC as an alternative solution to moteNIC. With those components, current TinyOS applications can be smoothly migrated from legacy MAC in TinyOS to SMAC.



Future plan includes:

a) Measure the throughput and latency of our stack with S-MAC.

b) Performance analysis of S-MAC in low-duty-cycle mode and in fully active mode.

c) Submit a journal paper on S-MAC with new implementation and measurement results.

4. Testbed Development

In this quarter, Wei Ye has worked with Mohammad Rahimi, Thanos Stathopoulos, Jerry Zhao and Naim Busek on testbed development at ISI. Our goal is to deploy about 20 PC/104s with Mica motes as a radio communication interface (moteNIC). Since our first application is for people detection and tracking in the ISI building, we also investigated how to use the COTS passive infrared (PIR) motion detector as the major senor on our testbed. We currently have 6 nodes deployed.

On the hardware side, Wei, Mohammad and Naim have worked to find and compare different PIR sensors and investigated how to interconnect the PIR sensors to PC/104 and Motes. Mohammad designed a power board that can be used to power the motes from the power source of the PC/104. Naim did further improvement on it for time synchronization and the PIR sensor connection.

Wei, Jerry, Fred and Yan worked together to replace the old plastic enclosures on PC/104s with custom-designed metal enclosures. The new enclosure solution not only provides better protection for the nodes in rigid environments, but also provides good shielding for CPU boards, reducing interference to RFM radio transceiver on moteNIC.

On the software side, Wei, Thanos and Jerry have worked to develop software for both motes and PC/104s as well as to upgrade the configurations of PC/104s. Beside the new communication stack, Wei also wrote code for motes to work with PIR sensors. Thanos ported the moteNIC code, originally developed by Jermey Elson, to our new communication stack.

Jerry has worked with Wei to upgrade the operating system and programming environment on the PC/104s. New features include: 1) Upgrading the Linux kernel from version 2.4.7 to 2.4.17 with devfs (device file system) support, which is essential for running moteNIC on PC/104s. 2) Upgrading system libraries and adding new management scripts to ease experimentation on the testbed. 3) Providing an integrated environment for applications on PC/104s to use the moteNIC and the new communication stack on motes.

Future plans for testbed development include:

a) More tests on the new testbed to reduce uncertainty in node deployment by measuring and monitoring the connectivity among nodes.

b) Deploy more nodes to form a large network.

c) Fabio Silva and Fred Stann plan to use the testbed to run diffusion, nested-queries, applications, and reliable transport protocols.

5. SensIT Collaboration/Integration

Fabio Silva and John Heidemann worked with Jim Reich (PARC), integrating Diffusion Routing and IDSQ. As we increase the number of applications running on top of Diffusion, we gain more experience on application requirements, allowing us to improve our algorithms and APIs. As we finish this integration task, we plan to present a joint demo to the Sensit community.

Also, during this quarter, Fabio Silva ran various experiments on both our Sensoria nodes at ISI as well as on the SensIT testbed at BAE in Austin, TX. In these experiments, we tried to quantify the performance of a few diffusion variants, including Diffusion, Diffusion + PUSH, Diffusion + GEAR, and Diffusion + PUSH + GEAR. Data collected from some of these experiments was shown to Sri Kumar during a demo in late August.

Fabio Silva and John Heidemann have continued to interact with other SensIT participants involving the diffusion software and its use by other projects in the program. Discussions this quarter included PARC, BBN, BAE, Auburn, Virginia Tech, U. Wisconsin, Cornell, and other SensIT groups.

The project set up the diffusion-users@isi.edu mailing list to encourage discussion of between diffusion users and developers. This list will support users of both native diffusion and diffusion running inside ns-2.

During the next quarter we plan to finish support integrating PARC's IDSQ with Directed Diffusion, presenting a joint demo to the SensIT community during the next PI meeting in November 2002.

6. Administrative

6.1 Staffing

As the current funding is winding down, we have greatly reduced SCADDS staffing. Graduate students Chalermek Intanagonwiwat and Ya Xu have left the project. Researcher Wei Ye and graduate students Jerry Zhao and Yen Yu are now funded by other projects and thus are beginning to shift their focus to other research. PIs Deborah Estrin and John Heidemann have dramatically scaled back the time they can devote to this research. Current efforts are primarily devoted to funding Fabio Silva at partial time to support continuing SensIT integration efforts. At projections of the current burn rate, this involvement must end by end-of November or December.

6.2 Travel

On June 10-12 John Heidemann attended the MOBIHOC conference to discuss developments in wireless networking and diffusion with the ad hoc networking community.

On June 13 John Heidemann gave a presentation "Wireless Sensor Networking Research at ISI" at Datalogisk Institute, University of Copenhagen. This talk was part of continuing collaboration between ISI and Philippe Bonnet (now at DIKU). The DIKU researchers are using diffusion filters to distribute database computation across a sensor network.

On 27 June, Wei Ye attended the IEEE INFOCOM 2002, and presented his paper "An energy-efficient MAC protocol for wireless sensor networks" in New York.

On 21-23 Aug. John Heidemann, Deborah Estrin, and Ramesh Govindan attended SIGCOMM. SIGCOMM discussions included a larger role of wireless and sensor networking in future SGICOMM conferences.

On 5-6 Sep. 2002 John Heidemann attended the IEEE CAS Workshop on Wireless Communications and Networking in Pasadena. He co-chaired the technical program with Mani Srivastava of UCLA.

On 23-24 Sep. 2002 John Heidemann and Fabio Silva went to PARC in San Jose to participate in collaborative experiments with Feng Zhao and Jim Reich's group.

6.3 Publications

Accepted this quarter:

Nirupama Bulusu, Vladimir Bychkovskiy, Deborah Estrin, and John Heidemann. Scalable, Ad Hoc Deployable, RF-Based Localization. In Proceedings of the Grace Hopper Celebration of Women in Computing, p. to appear. Vancouver, British Columbia, Canada, Institute for Women and Technology. October, 2002.
http://www.isi.edu/~johnh/PAPERS/Bulusu02a.html

Deepak Ganesan, Deborah Estrin, and John Heidemann. DIMENSIONS: Why do we need a new Data Handling architecture for Sensor Networks?. In Proceedings of the ACM Workshop on Hot Topics in Networks, p. to appear. Princeton, NJ, USA, ACM. October, 2002. http://www.isi.edu/~johnh/PAPERS/Ganesan02c.html.

Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed Diffusion for Wireless Sensor Networking. "ACM/IEEE Transactions on Networking", p. to appear, 2002. Pre-print of accepted ToN journal paper. http://www.isi.edu/~johnh/PAPERS/Intanagonwiwat02b.html.

Appeared this quarter:

John Heidemann, Fabio Silva, Yan Yu, Deborah Estrin, and Padmaparma Haldar. Diffusion Filters as a Flexible Architecture for Event Notification in Wireless Sensor Networks. Technical Report N. ISI-TR-556, USC/Information Sciences Institute, April, 2002. http://www.isi.edu/~johnh/PAPERS/Heidemann02a.html.

Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In Proceedings of the 22nd International Conference on Distributed Computing Systems, p. to appear. Vienna, Austria, IEEE. July, 2002. See UCLA CSD TR-01-750 for an expanded version of this paper. http://www.isi.edu/~johnh/PAPERS/Intanagonwiwat02a.html.

Wei Ye, John Heidemann, Deborah Estrin, "An energy-efficient protocol for wireless sensor networks," in Proceedings of the IEEE INFOCOM 2002, New York, June 2002.

Wei Ye, John Heidemann, Deborah Estrin, "A flexible and reliable radio communication stack on Mica motes." http://www.isi.edu/scadds/papers/commstack.pdf

Submitted this quarter:

Mohammad Rahimi, Hardik Shah, Gaurav Sukhatme, John Heidemann, and Deborah Estrin. Energy Harvesting in Mobile Sensor Networks. Submitted to ICRA. September, 2002.

6.4 Software released this quarter:

During this quarter, Fabio Silva released two new versions of our Directed Diffusion software to the SensIT community (Diffusion 3.1.1 on June 27th and Diffusion 3.1.2 on August 28th). New features include support for the updated Sensoria RF API, an update to GEAR (our Geographical and Energy Aware Routing protocol), which now supports PUSH; additional support for logging incoming and outgoing traffic using an i/o-level logging layer; and several API changes that allow more flexibility when using filters in Diffusion.

The software can be downloaded from the BBN website (http://dstl.bbn.com) or at the SCADDS website (http://www.isi.edu/SCADDS).

Wei Ye released the initial version of the radio communication stack on the Mica motes, along with a detailed documentation describing the design, implementation, APIs and some test results. The stack includes a physical layer and a MAC layer (S-MAC) implementation. We have received some positive feedback on its flexibility to allow different protocols to be developed in parallel. The software can be downloaded at http://www.isi.edu/scadds/software/motes/

The detailed documentation about the stack is available at
http://www.isi.edu/scadds/papers/commstack.pdf